REPO

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1992	Professional paper

4. TITLE AND SUBTITLE

PERFORMANCE MEASURES OF THE ADA RENDEZVOUS

5. FUNDING NUMBERS

PR: ZF01
PE: 0602936N
WU: DN300189

6. AUTHOR(S)

A. Sterrett and M. Minei

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Command, Control and Ocean Surveillance Center (NCCOSC),
Research, Development, Test and Evaluation Division (NRaD)
San Diego, CA 92152-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Chief of Naval Research
Independent Exploratory Development Programs (IED)
OCNR-20T
Arlington, VA 22217

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Ada Programming Language [ADA], was designed to meet the need for a standard computer programming language. Ada has the ability to take advantage of multiprocessor environments. One feature, known as the rendezvous, allows tasks to synchronize in such a environment. In this paper, we will look at the performance of the Ada rendezvous in a two-processor system.

In a distributed Ada system, the rendezvous provides synchronized communication between asynchronous tasks. A system of this sort would consist of at least two processors, each serving various tasks. Presently, the performance behavior of tasks which synchronize and possibly communicate are not widely known. Developers of concurrent Ada systems will need to perform sensitivity studies of the Ada tasking environment to develop efficient time-critical programs. Therefore, we will discuss Rendezvous Response Time from the point of view of a sensitivity study. We will show generalized performance curves of the rendezvous along with commentary on their performance elbows (or bottlenecks). Rendezvous Response Time will be defined as the amount of time one task must wait until its rendezvous request to another task is completed. The discussion will be based on two separate computer simulations of a two-processor system. The tools to be used to build these simulations are [ELSIR] and [APOSL].

Published in *Ada Letters*, Mar/Apr 92.

14 SUBJECT TERMS	15 NUMBER OF PAGES
ADA rendezvous nets	16 PRICE CODE

17 SECURITY CLASSIFICATION OF REPORT	18 SECURITY CLASSIFICATION OF THIS PAGE	19 SECURITY CLASSIFICATION OF ABSTRACT	20 LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

21a. NAME OF RESPONSIBLE INDIVIDUAL	21b. TELEPHONE *(Include Area Code)*	21c. OFFICE SYMBOL
A. Sterrett	(619) 553–4078	Code 411

PERFORMANCE MEASURES OF THE ADA RENDEZVOUS

Anthony Sterrett
Naval Ocean Systems Center
Code 411
San Diego, CA 92152
(619) 553-4078
sterrett@nosc.mil

Marvin Minei
Naval Ocean Systems Center
Code 411
San Diego, CA 92152
(619) 553-4094
minei@nosc.mil

I. ABSTRACT

The Ada Programming Language [ADA], was designed to meet the need for a standard computer programming language. Ada has the ability to take advantage of multiprocessor environments. One feature, known as the rendezvous, allows tasks to synchronize in such a environment. In this paper, we will look at the performance of the Ada rendezvous in a two-processor system.

In a distributed Ada system, the rendezvous provides synchronized communication between asynchronous tasks. A system of this sort would consist of at least two processors, each serving various tasks. Presently, the performance behavior of tasks which synchronize and possibly communicate are not widely known. Developers of concurrent Ada systems will need to perform sensitivity studies of the Ada tasking environment to develop efficient time-critical programs. Therefore, we will discuss Rendezvous Response Time from the point of view of a sensitivity study . We will show generalized performance curves of the rendezvous along with commentary on their performance elbows (or bottlenecks). Rendezvous Response Time will be defined as the amount of time one task must wait until its rendezvous request to another task is completed. The discussion will be based on two separate computer simulations of a two-processor system. The tools to be used to build these simulations are [ELSIR] and [APOSL].

II. PERFORMANCE MEASURES FOR A TRANSACTION BASED SYSTEM

We present a discussion on the average Rendezvous Response Time for the following transaction based two-processor system. (By a transaction based system, we mean a system in which there is a continuous stream of customers arriving for service from the system. When each customer has completed its service requirements, it departs from the system completely.) Rendezvous Response Time will be defined as the amount of time one task must wait until its rendezvous request to another task is completed. The discussion will be based on data collected from a computer simulation of the two-processor system.

We will assume all arrival rates of customers to the model are Poisson distributed with appropriate average arrival rate parameters and that their service demands on the processors are exponentially distributed with appropriate average service demand parameters.

From here on, the word "average" will be assumed where appropriate.

A. DESCRIPTION OF THE TRANSACTION BASED SYSTEM

Consider a two-processor system as drawn in Figure 1. Task SERVER_TASK will execute exclusively on one processor that we will call Server Processor. SERVER_TASK will have the following form.

```
task body SERVER_TASK is
begin
        loop
          accept Data_Exchange (. . .) do
          . . .
          end Data_Exchange;
        end loop;
end SERVER_TASK;
```

SERVER_TASK will execute as a reentrant task, servicing one rendezvous request made to its "Data_Exchange" entry each time it gains access to the processor. This service will be done in a first-come, first-served manner. Also, there will be other tasks that arrive at Server Processor at some average rate and will provide contention with SERVER_TASK for processor resources. These tasks play no role in the rendezvous itself and will be called the "traffic tasks" of Server Processor. Each of these tasks will require only a finite amount of service time before exiting the entire model. SERVER_TASK will enter and remain in the SERVER_TASK delay server whenever there are no rendezvous requests to service.

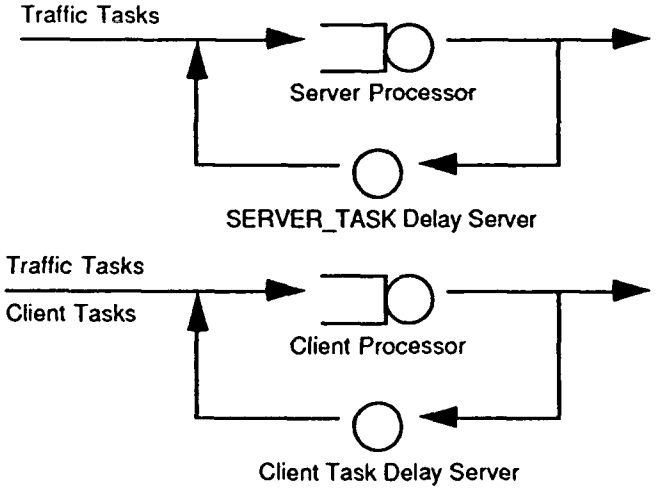

Figure 1. The transaction based two-processor system.

At the second processor, other tasks will arrive at some given rate. These tasks will make rendezvous requests to SERVER_TASK. Thus, we will refer to these tasks as "client tasks" and this processor as Client Processor. In this case, client tasks will have the form of CLIENT_TASK as defined below.

```
task body CLIENT_TASK is
begin
        . . .
        SERVER_TASK.Data_Exchange (. . .);
        . . .
end CLIENT_TASK;
```

A client task arrives at Client Processor and enters the processor queue. When it is selected for service, the client task makes a rendezvous request with SERVER_TASK and is put in a "blocked" state (i.e., a state where the client task is no longer allowed to continue execution on its processor) and enters the rendezvous delay server. At this point, the Client Processor is assigned to another task in the queue. Client tasks that enter into the rendezvous delay server will remain there until their rendezvous request is serviced by the SERVER_TASK. The amount of time that a client task spends in the rendezvous delay server is its Rendezvous Response Time. Also at Client Processor, tasks that play no role in the rendezvous will arrive for service. We refer to them as the "traffic tasks" of Client Processor. It should be noted that the "traffic tasks" of Server Processor and Client Processor are two different customer classes of the model.

By assuming the Forced-Flow Law, the arrival rate of client tasks to Client Processor is equal to the arrival rate of rendezvous requests to the "Data_Exchange" entry queue. A consequence of this assumption is that the Rendezvous Response Time will only be dependent on the arrival rate of client tasks to Client Processor, the arrival rate of traffic tasks of Server Processor, the service demands of these traffic tasks on Server Processor, and the service demand of SERVER_TASK on Server Processor.

B. COMPUTER SIMULATION RESULTS

Figure 2 are general curve drawings of the Rendezvous Response Time based on simulated data of the transaction based system. For these curves, U_R is defined as the Server Processor utilization devoted to the execution of SERVER_TASK working to complete each rendezvous and U_H is defined as the Server Processor utilization devoted to the execution of its traffic tasks. It can be analytically shown that

$$U_R = \left(\begin{array}{l}\text{Arrival rate of client tasks}\\ \text{to Client Processor}\end{array}\right) * \left(\begin{array}{l}\text{Service demand of SERVER\_TASK}\\ \text{at Server Processor}\end{array}\right)$$

$$U_H = \left(\begin{array}{l}\text{Arrival rate of traffic tasks}\\ \text{to Server Processor}\end{array}\right) * \left(\begin{array}{l}\text{Service demand of traffic tasks}\\ \text{at Server Processor}\end{array}\right)$$

(Recall that the arrival rate of client tasks to the Client Processor is equal to the arrival rate of rendezvous requests to the "Data_Exchange" entry queue.)

Throughout Figure 2, all service demands are fixed with the arrival rates varied to obtain the utilities. Each curve is sketched as a function of U_H with U_R fixed at the following values.

A. For Curve (1), U_R is fixed at 60%.

B. For Curve (2), U_R is fixed at 40%.
C. For Curve (3), U_R is fixed at 20%.
D. For Curve (4), U_R is fixed at 10%.
E. For Curve (5), U_R is fixed at 5%.

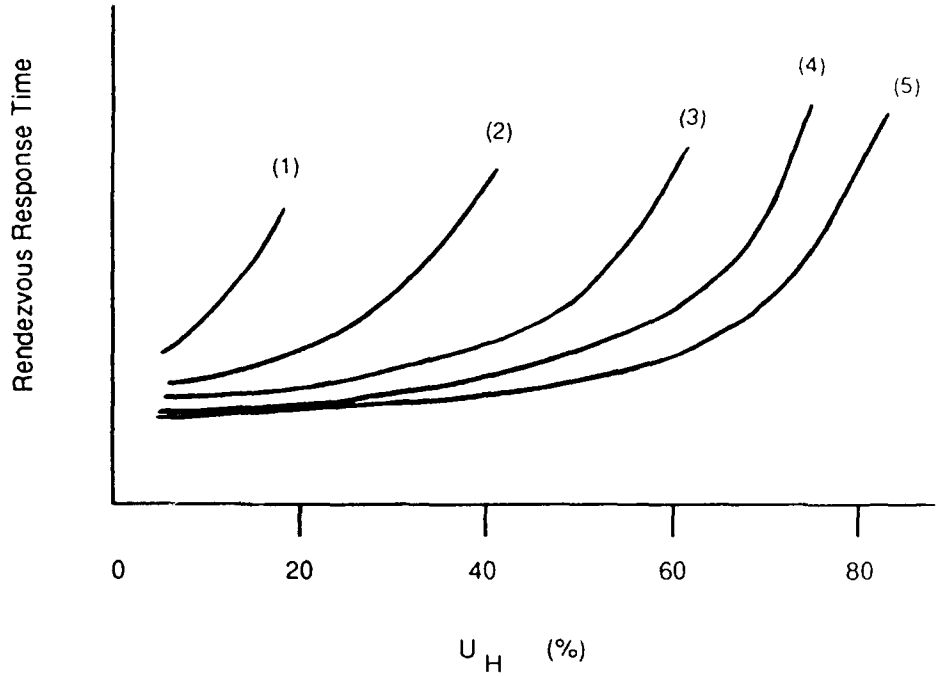

Figure 2. Graph of the Rendezvous Response Time as determined from simulated data of the transaction based system.

C. DISCUSSION OF RESULTS

Consider the curves of Figure 2 that correspond to small values of U_R. A noticeable "elbow" appears in them at various values of U_H. To the left of the elbow, the curve increases slowly. To the right of the elbow, the curve goes to infinity at an accelerated rate. The appearance of the elbows occur at decreasing values of U_H as we increase U_R. At large enough values for U_R, the elbow appearance is immediate.

Analysis of the curves show that in the region to the left of the elbow, the queueing effects due to rendezvous requests at the "Data_Exchange" entry have a minimal impact on the Rendezvous Response Time. Thus for this region of U_H, the Server Processor basically serves both its traffic tasks and the SERVER_TASK as though the traffic tasks and rendezvous requests were arriving directly at the Server Processor queue for service. Such information could be used for bounding the arrival rates and service demands of customers entering a two-processor system. Within these bounds, the effects of the rendezvous will have a small effect on the system response time for the client tasks. Outside of these bounds, an explosion in response time for the client tasks will occur.

III. ACKNOWLEDGEMENTS

This work was funded by the Naval Ocean Systems Center's Independent Exploratory Development Program.

IV. REFERENCES

(1) [ADA] United States Department of Defense, "Reference Manual for the Ada Programming Language," ANSI/MIL-STD-1815A, 1983. (Ada is a registered trademark of the U.S. Government.)

(2) [APOSL] Naval Ocean Systems Center, "Ada Process Oriented Simulation Library," NOSC TM-641, 1991.

(3) [ELSIR] ADV Technologies, ELSIR (Evaluation des Systemes Informatiques Repartis).